UNIT-1

Software Development Life Cycle (SDLC)

SDLC stands for Software Development Life Cycle, which is a process that software development organizations use to plan, design, build, test, and deliver software applications. The SDLC process outlines the steps and stages involved in creating a software product, from the initial planning phase to the final deployment and maintenance phase. The goal of the SDLC is to ensure that the software is of high quality, meets the requirements of the end-users, and is delivered on time and within budget. The SDLC process provides a structured and systematic approach to software development that helps to minimize risks and ensure that the end result is a high-quality software product.

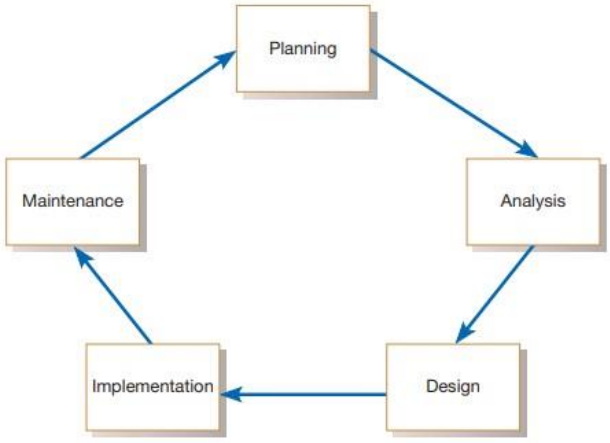

Fig:Phases of SDLC

1. Planning
   The first phase in the SDLC is planning. In this phase, someone identifies the need for a new or enhanced system. In larger organizations, this recognition may be part of a corporate and systems planning process. Information needs of the organization as a whole are examined, and projects to meet these needs are proactively identified. The organization's information system needs may result from requests to deal with problems in current procedures, from the desire to perform additional tasks, or from the realization that information technology could be used to capitalize on an existing opportunity. In smaller organizations, determination of which systems to develop may be affected by ad hoc user requests submitted as the need for new or enhanced systems arises, as well as from a formalized information planning process. It is further divided into two sub-phases:
   i. Project Identification and Selection
      During project identification and selection, an organization determines whether resources should be devoted to the development or enhancement of each information system under consideration. The outcome of the project identification and selection process is a determination of which systems development projects should be undertaken by the organization.
   ii. Project Initiation and Planning
      Project initiation focuses on activities designed to assist in organizing a team to conduct project planning. Project planning focuses on defining clear,discrete activities and the work needed to complete each activity within a single project.

2. Analysis

   The second phase in the SDLC is analysis. During this phase, the analyst thoroughly studies the organization's current procedures and the information systems used to perform organizational tasks. Analysis has two subphases. The first is requirements determination. In this subphase, analysts work with users to determine what the users want from a proposed system. The requirements determination process usually involves a careful study of any current systems, manual and computerized, that might be replaced or enhanced as part of the project. In the second part of analysis, analysts study the requirements and structure them according to their interrelationships and eliminate any redundancies. The output of the analysis phase is a description of the alternative solution recommended by the analysis team.

3. Design

   The third phase in the SDLC is design. During design, analysts convert the description of the recommended alternative solution into logical and then physical system specifications. The analysts must design all aspects of the system, from input and output screens to reports, databases, and computer processes. The analysts must then provide the physical specifics of the system they have designed, either as a model or as detailed documentation, to guide those who will build the new system. That part of the design process that is independent of any specific hardware or software platform is referred to as logical design. Theoretically, the system could be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended. Logical design concentrates on the business aspects of the system and tends to be oriented to a high level of specificity. Once the overall high-level design of the system is worked out, the analysts begin turning logical specifications into physical ones. This process is referred to as physical design. As part of physical design, analysts design the various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output.

4. Implementation

   Implementation includes coding, testing, installation, documentation, training and support. The physical system specifications, whether in the form of a detailed model or as detailed written specifications, are turned over to programmers as the first part of the implementation phase. During implementation, programmers turn system specifications into a working system that is tested and then put into use. During coding, programmers write the programs that make up the system. Sometimes the code is generated by the same system used to build the detailed model of the system. During testing, programmers and analysts test individual programs and the entire system in order to find and correct errors. During installation, the new system becomes part of the daily activities of the organization. Application software is installed, or loaded, on existing or new hardware, and users are introduced to the new system and trained.

5. Maintenance

   When a system is operating in an organization, users sometimes find problems with how it works and often think of better ways to perform its functions. Also, the organization's needs with respect to the system change over time. In maintenance, programmers make the changes that users ask for and modify the system to reflect evolving business conditions. These changes are necessary to keep the system running and useful. In a sense, maintenance is not a separate phase but a repetition of the other life cycle phases required to study and implement the needed changes.

There are 4 main activities performed in this phase:

a) Obtaining maintenance requests
b) Transforming requests into changes
c) Designing changes
d) Implementing changes

Phase	Products, Outputs, or Deliverables
Planning	Priorities for systems and projects; an architecture for data, networks, and selection hardware, and information systems management are the result of associated systems
	Detailed steps, or work plan, for project
	Specification of system scope and planning and high-level system requirements or features
	Assignment of team members and other resources
	System justification or business case
Analysis	Description of current system and where problems or opportunities exist, with a general recommendation on how to fix, enhance, or replace current system
	Explanation of alternative systems and justification for chosen alternative
Design	Functional, detailed specifications of all system elements (data, processes, inputs, and outputs)
	Technical, detailed specifications of all system elements (programs, files, network, system software, etc.)
	Acquisition plan for new technology
Implementation	Code, documentation, training procedures, and support capabilities
Maintenance	New versions or releases of software with associated updates to documentation, training, and support

Table: Deliverables of SDLC phases

System Development Methodology or Software Process Model

It is standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

Traditional Waterfall Model of System Development

The waterfall model is a linear, sequential approach to the software development lifecycle (SDLC) that is popular in software engineering and product development. The waterfall model uses a logical progression of SDLC steps for a project, similar to the direction water flows over the edge of a cliff. It sets distinct endpoints or goals for each phase of development. Those endpoints or goals can't be revisited after their completion.

Advantages of waterfall model: Simple method and easy to use, phases are clear, suitable for smaller projects, Easy to manage

Disadvantages of waterfall model: Doesn't allow much revision, not suitable for complex projects, doesn't include a feedback path, customer can see working model of the project only at the end

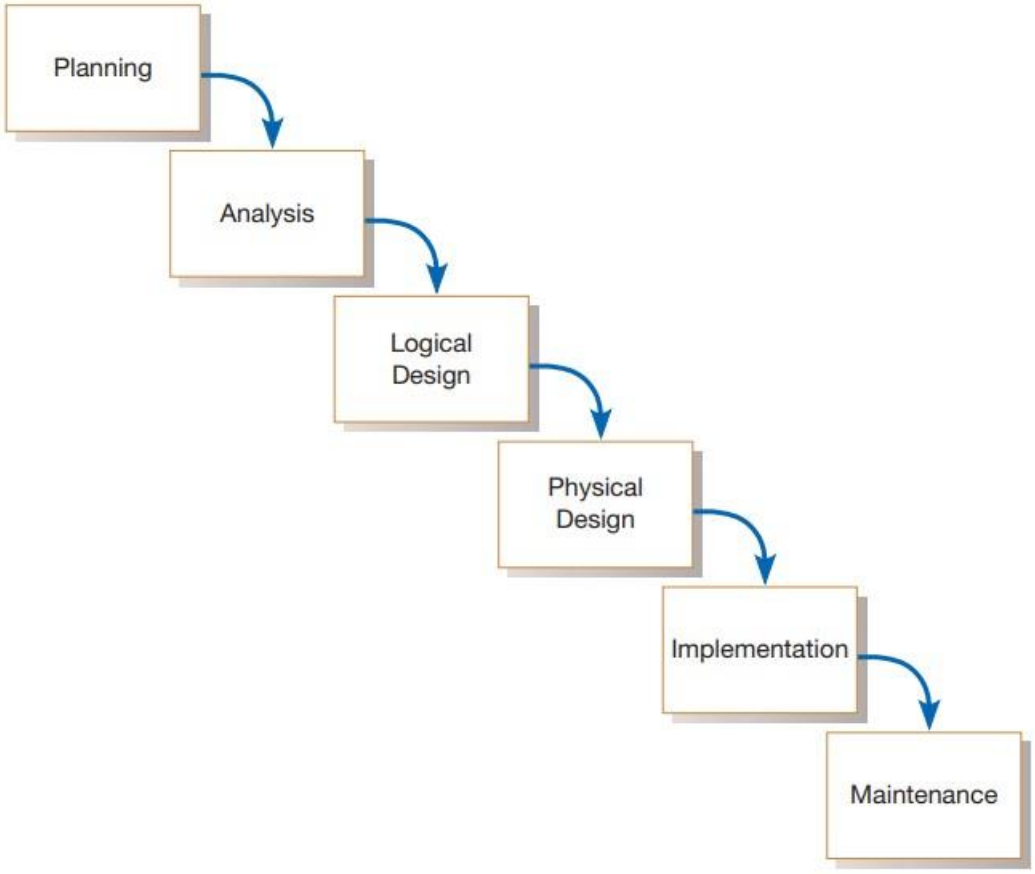

Fig: Waterfall model

Spiral Model of System Development

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model. The spiral model incorporates the stepwise approach of the Classical Waterfall Model. The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique. Also, the spiral model can be considered as supporting the Evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of spiral model

1. Risk Handling
2. Good for large projects
3. Flexibility in Requirements
4. Customer Satisfaction

Disadvantages of spiral model

1. Complex
2. Expensive
3. Too much dependability on Risk Analysis
4. Difficulty in time management

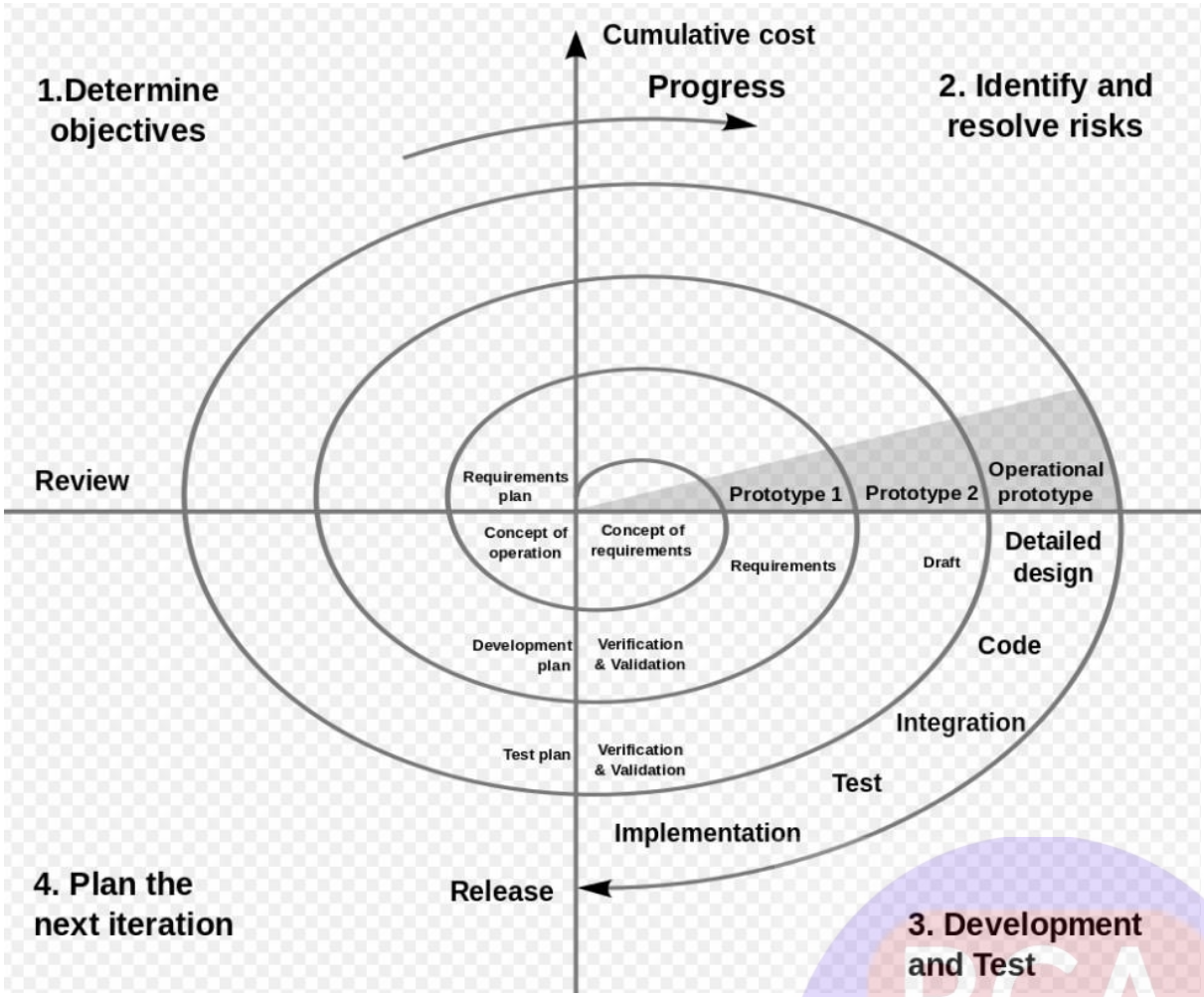

Fig: Spiral Model

Agile Methodology

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly. Agile project management is not a singular framework — rather, it can be used as an umbrella term to include many different frameworks. Agile project management can refer to terms including Scrum, Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF).

Agile's four main values are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

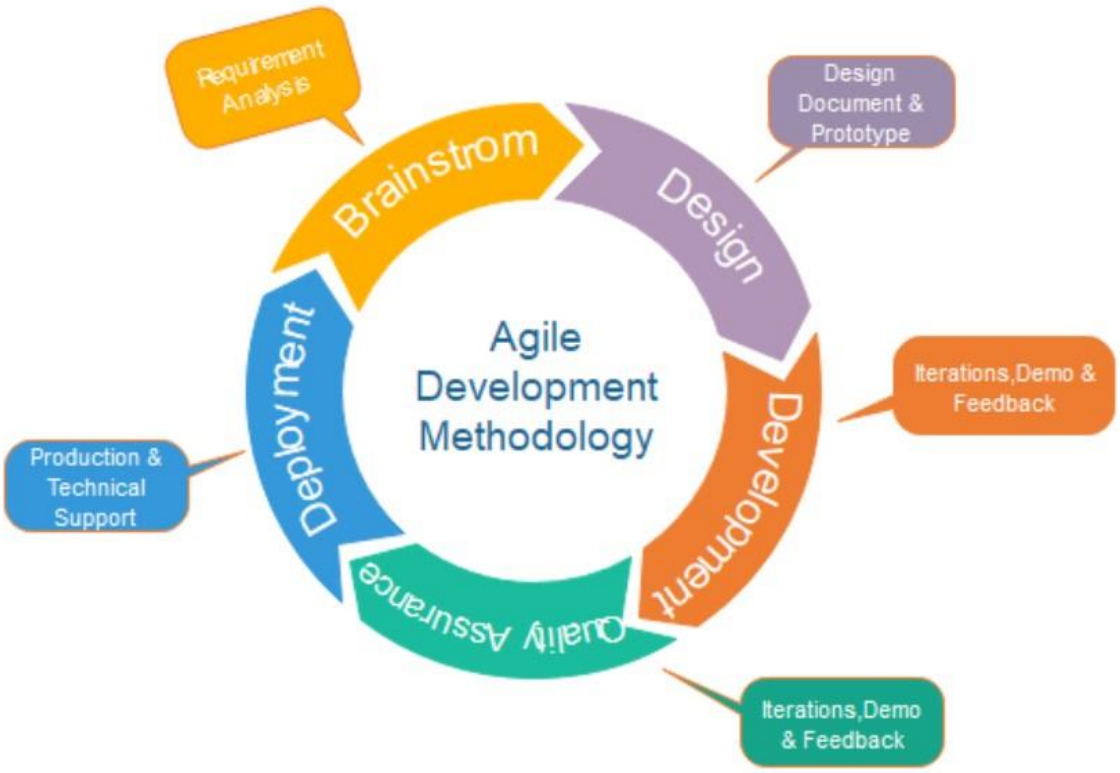

Fig. Agile Model

eXtreme Programming

An approach to software development known as "extreme programming" (XP) aims to increase software responsiveness and quality. It promotes frequent releases in quick development cycles as a sort of agile software development, with the goal of increasing productivity and introducing checkpoints where new customer requirements can be accepted. Other characteristics of extreme programming include working in pairs or performing extensive code reviews, unit testing all code, delaying the development of features until they are actually required, a flat organizational structure, simple and clear code, anticipating changes in the customer's requirements over time as the issue is better understood, and regular interaction between programmers and the client. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels.

Advantages of XP: stable system, clear code, less documentation, customer satisfaction

Disadvantages of XP: Unclear estimates, stressful, pair programming takes longer

CASE tools

They are software tools that provide automated support for some portion of the systems development process. CASE tools are used to support a wide variety of SDLC activities. CASE tools can be used to help in multiple phases of the SDLC: project identification and selection, project initiation and planning, analysis, design, and implementation and maintenance.

The general types of CASE tools are listed below:

• Diagramming tools enable system process, data, and control structures to be represented graphically.

• Computer display and report generators help prototype how systems "look and feel." Display (or form) and report generators make it easier for the systems analyst to identify data requirements and relationships.

• Analysis tools automatically check for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports.

• A central repository enables the integrated storage of specifications, diagrams, reports, and project management information.

• Documentation generators produce technical and user documentation in standard formats.

• Code generators enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports

SDLC Phase	Key Activities	CASE Tool Usage
Project identification and selection	Display and structure high-level organizational information	Diagramming and matrix tools to create and structure information
Project initiation and planning	Develop project scope and feasibility	Repository and documentation generators to develop project plans
Analysis	Determine and structure system requirements	Diagramming to create process, logic, and data models
Logical and physical design	Create new system designs	Form and report generators to prototype designs; analysis and documentation generators to define specifications
Implementation	Translate designs into an information system	Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation
Maintenance	Evolve information system	All tools are used (repeat life cycle)

Differentiate between agile methodology and waterfall model.

Agile Methodology	Waterfall Model
It takes iterative approach to software development.	It takes sequential approach to software development.
It works well when scope of project is unknown.	It works well when the scope of project is known beforehand.
It allows changes to be made even after the development starts.	The contract terms wont allow changes to. be made as the proces is sequential.
Customer availability is important throught the entire project.	Customer availability is required only while setting milestone.
Provides flexibility to oversee the development of project.	flexibility is limited.
Products are tested frequently for bugs and errors during develop- ment cycle.	Testing can't be done during development cycle, but only at the end.

Information System

Information System (IS) is a set of interrelated components that creates, processes, stores, retrieves and disseminates information to facilitate organizational decision-making process. Information systems also help managers and workers analyze problems, visualize complex subjects, and create new products. The input to such a system is data and processed data becomes information. Information systems are guided by a set of policies, principles procedures and resources.

Types of Information System:

1. Transaction Processing System [TPS]:
   A transaction processing system is a computerized system that performs and records the daily routine transactions necessary to conduct business, such as sales order entry, hotel reservations, payroll, employee record keeping, and shipping. TPS meets the information needs of operational managers. The major purpose of TPS is to answer routine questions and to track the flow of transactions through the organization.It is source of data to MIS and DSS.
2. Management Information Systems [MIS]:
   MIS is a specific category of information systems serving middle management. MIS provide middle managers with reports on the organization's current performance. This information is used to monitor and control the business and predict future performance. MIS summarize and report on the company's basic operations using data supplied by transaction processing systems. The basic transaction data from TPS are compressed and usually presented in reports that are produced on a regular schedule. MIS are general purpose and well-integrated systems that meet the tactical information needs of middle managers.
3. Decision Support Systems (DSS):
   Decision support systems are interactive computer based systems, which help decision makers utilize data and models to solve semi-structured problems. A DSS is the type of intelligent support system that integrates internal and external data with various decision-making models in order to produce alternative solutions to a given problem. It helps to automate routine and repetitive elements in a problem while simultaneously supporting the use of intuition and judgments.
4. Executive support systems(ESS):
   It is primarily used by top level management. It is user friendly, interactive system, designed to meet information needs of top management engaged in long-range planning, crisis management, and other strategic decision(unique, non- repetitive and future oriented), which addresses long-term issues such as emerging markets, merger and acquisition strategies, new product development and investment strategies. Thus, Executive information systems are intended to be used by the senior managers directly to provide support to non- programmed decisions in strategic management. ESS assists in the making of decision that requires an in-depth understanding of the firm and of the industry in which the firm operates
5. Expert System (ES):
   Expert system is an artificial intelligence based system that converts the knowledge, experience, intuition and judgment of human expert to help organizations acquire and retain knowledge that is vital to the competitiveness and success of the company. Expert systems are good at solving semi-structured and unstructured problems and can solve complex problems that require theoretical knowledge and practical experts.
   Expert systems typically consist of three parts:
   (1) a knowledge base which contains the information acquired by interviewing experts, and logic rules that govern how that information is applied;

(2) an Inference engine that interprets the submitted problem against the rules and logic of information stored in the knowledge base;
(3) user interface that allows the user to express the problem in a human language such as English.

6. Office Automation System(OAS)

Office automation systems are designed to increase the productivity of clerical & knowledge workers and enhance communication in the workplace. Office automation uses software and hardware solutions to ease your workload. Varieties of office automation systems are now applied to business and communication functions that used to be performed manually or in multiple locations of a company, such as preparing written communications and performing regular official activities.

Differentiate between Gantt chart and PERT diagram.

Gantt Chart	**PERT Diagram**
It shown duration of tasks.	It usually shows dependencies between tasks.
It shows time overlap between tasks.	It shows tasks that can be done in parallel.
It is employed for small projects.	It is used in large and complex projects.
Time duration and percentage completion have accurate values.	Completion time is decided based upon prediction or past processing.
Visually shows stack time in schedule.	Show stack time by data in rectangles.

Differentiate between structured and object oriented methodologies.

Object oriented methodologies	**Structured methodologies**
The main focus is on data and real world objects that are important.	The main focus is on process and procedures of a system.
It uses incremental or iterative methodology.	It uses SDLC methodology.
The techinque is new and mostly preferred.	This techinque is old and not usually preferred.
Suitable for projects with changing user requirements.	Suitable for projects with well defined user requirements.
Low risk and high reusability.	High risk and low reusability.

UNIT-2:Planning

Project Identification and Selection, Project Initiation and Planning

Project Manager

Activities performed by project manager during initiation, planning, execution and termination of project are:

1. Initiating the project. During project initiation, the project manager performs several activities.
   a) Establishing relationship with customer
   b) Establishing project initiation plan and management procedures.
   c) Establishment of project initiation team
   d) Establishing project management environment & workbook
   e) Developing project charter.
2. Planning the project: Activities performed while planning the project are:
   a) Describing project scope, alternatives and feasibility
   b) Dividing project into managable tasks.
   c) Estimating resources and creating a resource plan
   d) Developing Preliminary schedule communication plan
   e) Identifying and assessing risk
3. Executing the project: Following activities are performed during project execution.
   a) Executing baseline project plan
   b) monitoring project progress
   c) Managing changes to baseline project plan
   d) maintaining project workbook
   e) Communicating project status.
4. Closing down the project: The activities performed during this phase are:
   a) Termination of project.
   b) conducting post project reviews
   c) closing customer contract.

Corporate Strategic Planning (CSP)

Corporate strategic planning is an ongoing process that defines the mission, objectives and strategies of an organization. It's outcomes are:

1. Mission Statement: A statement that makes it clear what business a company is in.
2. Statement of Objective: A series of statement that express on organization's qualitatuve and quantitative goals for reaching a desired future position.
3. Competitive Strategy: The method by which an organization attempts to achieve its mission and objective. The generic competitive strategies are low cost producer, product differentiation and product focus or niche.

Corporate information system planning (CISP)

Corporate information system planning is an orderly means of assessing the information needs of an orgainzation and defining the system's databases and technologies that will best satisfy those needs.

It involves the following activities:

1. Describe the current situation
    a. Top down planning
    b. Bottom up planning
    c. Planning team is chartered to model existing situation
    d. Matrices are developed to cross reference units
2. Describe the target situation
    a. Update list of orgainzational locations, functions to reflect described locations,functions
    b. Matrices are updated to reflect the future states
    c. Planners focus on difference between current lists and future lists and matrices
3. Developing transition strategy and plans
    a. Broad comprehensive documents are created that looks at both short term and long term organization development needs
    b. Consists a series of project

Project charter

A project charter is a short document that is prepared for both internal and external stakeholders. It provides high level overview of the project. It is a useful communication tool that helps to assure that the organization and other stakeholders understand the initiation of project/

It is used in project planning to develop Baseline Project Plan(BPP) and Project Scope Statement(PSS).

Baseline Project Plan(BPP)

It is a document that has all the information collected during project initiation and planning. Once the BPP is completed, a formal review of project plan can be conduted with project clients and other interested parties.

It has four major sections: Introduction, System Description, Feasibility Assessment, Management Issues.

Project Scope Statement

It is a short document prepared for the customer that describes what the project will deliver and outlines all the work required to complete the project. It ensures that both developer and customer gain common understanding of the project. It is very easy to create because it contains high level summary of BPP information.

Feasibility Analysis

Feasibility analysis is the process of evaluating the potential of a proposed project or venture. It is used to determine whether the project is viable, and to identify any potential issues that may arise during the implementation of the project. The analysis typically includes a review of technical, financial, legal, and operational aspects of the project, as well as an assessment of the potential risks and benefits. The goal of a feasibility analysis is to provide decision-makers with a clear understanding of the project's potential and to help them decide whether to proceed with the project or not.

Types of project feasibility:

1. Economic feasibility
   Economic feasibility is a type of feasibility analysis that assesses the potential financial performance of a proposed project or venture. The goal of economic feasibility analysis is to determine whether the project is financially viable, and to identify any potential financial risks or issues that may arise during the implementation of the project. Economic feasibility analysis typically includes a review of the project's costs, revenues, and profits. It also involves forecasting future financial performance using techniques such as break-even analysis, net present value (NPV), and internal rate of return (IRR). The analysis will also include identification of potential funding sources and consideration of the projects impact on the overall economy. The goal of economic feasibility analysis is to provide decision-makers with a clear understanding of the project's potential financial performance and to help them decide whether to proceed with the project or not, by comparing the project's expected benefits to its expected costs and evaluating the overall return on investment (ROI) of the project.
2. Techincal feasibility
   Technical feasibility is the assessment of whether a proposed project or solution can be implemented with the existing technology and resources. It involves determining if the necessary technology and resources are available and if they can be used effectively and efficiently to achieve the project's objectives. The technical feasibility assessment typically includes the following steps:
   i. Identifying the technical requirements of the project: This includes determining the hardware, software, and other resources that are required to implement the project.
   ii. Assessing the availability of the necessary technology and resources: This includes determining if the required technology and resources are currently available or if they need to be developed or acquired.
   iii. Evaluating the compatibility of the proposed solution with existing systems: This includes determining if the proposed solution can integrate with existing systems and if it is compatible with the existing infrastructure.
   iv. Estimating the costs and benefits of the proposed solution: This includes determining the costs associated with acquiring and implementing the technology and resources and comparing them to the benefits that the project is expected to generate.

v. Identifying potential technical risks and developing a risk management plan: This includes identifying potential technical issues that could arise during the implementation of the project and developing a plan to mitigate or address those risks.

3. Operational feasibility

Operational feasibility is the assessment of whether a proposed project or solution can be implemented and sustained within the existing organizational structure and resources. It involves determining if the proposed solution can be effectively and efficiently integrated into the day-to-day operations of the organization and if the necessary resources, including personnel and funding, are available to support the project. The operational feasibility assessment typically includes the following steps:

i. Identifying the organizational impact of the proposed solution: This includes determining how the proposed solution will impact the existing organizational structure, processes, and roles.

ii. Assessing the availability of the necessary personnel and resources: This includes determining if the necessary personnel and resources, such as funding and equipment, are available to support the project.

iii. Evaluating the compatibility of the proposed solution with existing organizational policies and procedures: This includes determining if the proposed solution is consistent with existing organizational policies and procedures and if it will require any changes to them.

iv. Identifying potential operational risks and developing a risk management plan: This includes identifying potential operational issues that could arise during the implementation of the project and developing a plan to mitigate or address those risks.

v. Establishing a support and maintenance plan: This includes determining how the proposed solution will be supported and maintained once it is implemented

4. Schedule feasibility

Schedule feasibility is the assessment of whether a proposed project or solution can be completed within the specified time frame. It includes identifying the project milestones and deadlines, assessing the availability of the necessary personnel and resources, evaluating the dependencies between tasks and activities, identifying potential schedule risks and developing a risk management plan, and establishing a project monitoring and control plan.

5. legal and contractual feasibility

Legal and contractual feasibility is the assessment of whether a proposed project or solution is compliant with the relevant laws, regulations, and contracts. It involves determining if the proposed project or solution complies with the legal and contractual requirements that may apply to it, such as environmental, health and safety, and data privacy regulations, and if it does not, what are the necessary steps to make it compliant.

6. Political feasibility

   Political feasibility is the assessment of whether a proposed project or solution is likely to be supported and approved by the relevant political stakeholders and decision-makers. It includes identifying the relevant political stakeholders, assessing the political priorities and agendas, evaluating the potential political risks, identifying the necessary funding and approvals, and building support and alliances.

Time Value of Money

The time value of money (TVM) is a concept that states that money in the present is worth more than the same amount of money in the future. This is because money in the present can be invested and earn interest, whereas money in the future is worth less due to inflation. TVM is used to calculate the present value (PV) of a future cash flow, which is the value of that cash flow today, considering the time value of money. TVM also used to calculate future value (FV) of a present cash flow, which is the value of that cash flow in the future, considering the interest rate.

The basic formula for calculating the present value (PV) is:

PV = FV / (1+r) ^t

Where FV is the future value, r is the interest rate, and t is the number of time periods.

The basic formula for calculating the future value (FV) is:

FV = PV * (1+r) ^t

Where PV is the present value, r is the interest rate, and t is the number of time periods.

Net present value (NPV)

Net Present Value (NPV) is a financial metric that calculates the present value of future cash flows generated by an investment, minus the initial cost of the investment. It is commonly used to evaluate the economic feasibility of a project or investment.

The formula for NPV is:

NPV = (CF1 / (1+r) ^1) + (CF2 / (1+r) ^2) + ... + (CFn / (1+r) ^n) - C0

Where: CF = cash flow, r = discount rate, n = the number of time periods, C0 = the initial cost of the investment.

Return on Investment (ROI)

Return on Investment (ROI) is a financial metric that measures the efficiency of an investment, by calculating the ratio of the net gain or loss from an investment to the initial investment cost. It is commonly used to evaluate the economic feasibility of a project or investment. The formula for ROI is:

ROI = (Net Gain or Loss from Investment / Initial Investment Cost) x 100

Break Even Analysis and Break Even Point

Break-even analysis is a financial tool used to determine the point at which the revenue of a project or investment equals its costs. It is commonly used to evaluate the economic feasibility of a project or investment. The break-even point is the point at which the total revenue equals the total cost, and it is represented by the intersection of the revenue and cost curves.

UNIT-3: Analysis

Requirements Determination, Requirements Structuring

Methods of Determining System Requirements

1. Traditional Methods of Collecting System Requirement
   • Individually interview people informed about the operation and issues of the current system and future systems needs
   • Interview groups of people with diverse needs to find synergies and contrasts among system requirements
   • Observe workers at selected times to see how data are handled and what information people need to do their jobs
   • Study business documents to discover reported issues, policies, rules, and directions as well as concrete examples of the use of data and information in the organization
2. Contemporary Methods for Collecting System Requirements
   • Bringing session users, sponsors, analysts, and others together in a JAD session to discuss and review system requirements
   • Using CASE tools during a JAD to analyse current systems to discover requirements that will meet changing business conditions
   • Iteratively developing system prototypes that refine the understanding of system requirements in concrete terms by showing working versions of system features
3. Radical methods of determining system requirements
   - Business process reengineering
     Business Process Reengineering (BPR) is a management approach that involves the radical redesign of core business processes to achieve dramatic improvements in performance. BPR is used to help organizations become more efficient and effective by streamlining processes, reducing costs, and improving customer service.

     BPR typically involves the following steps:
     1. Identifying the current business processes
     2. Analyzing the current processes
     3. Designing new processes:
     4. Implementing the new processes:
     5. Measuring the results:

   - Disruptive Technologies
     Disruptive technologies are innovations that fundamentally change the way industries or markets operate. These types of technologies often disrupt established businesses and create new opportunities for growth. Examples of disruptive technologies include the internet, mobile devices, and ride-sharing services.

Qualities and Skills of System Analyst

1. Communication Skills

System analysts spend a great deal of time engaging with users, consumers, management and developers according to the nature of the work. The performance of a project may rely on the system analyst clearly communicating information such as project specifications, adjustments required and results of testing. Fluent language skills and written communication abilities are important to succeed as a system analyst.

2. Technical Skills

An analyst should know what IT technologies are being used in order to find system solutions, what new potential results can be accomplished across existing systems, and what the latest technology offers. The system analyst must know the methods of system design.

3. Analytical Skills

The skill set of an analyst should include excellent analytical abilities in order to better analyse the needs of a client and convert them into application and organizational requirements. Analyzing information , records, user feedback surveys and workflow to decide which course of action can correct the system issue is one aspect of the job. Good analytical abilities are helpful in effectively conducting the work of the system analyst.

4. Problem-Solving Skills

Although analysts are not exceptional in the ability to develop workable solutions to problems, it is a required skill to effectively perform the job. As with most IT positions, the career of the system analyst can be spent coping with regular and random modifications. When these professionals are working to designing custom system solutions, nothing is 100 percent predictable – so finding ways to quickly fix problems and move toward a project 's efficient completion is critical in the system analyst 's position.

5. Decision-Making Skills

The capacity to make decisions is another significant system analyst skill. In a broad range of system problems, the system analyst is called upon for sound judgement as a management consultant and developer advisor, any number of which might decide the feasibility of the system. It should be possible for professionals who wish to pursue a career as a system analyst to analyse a situation, gain feedback from stakeholders, and choose a course of action.

6. Managerial Skills

The ability to manage projects is another ability a system analyst should possess. Planning the scope of the project, directing team members, managing demands for adjustment, predicting budgets and keeping everyone on the project under assigned time limits are just some of the management skills that a system analyst should have. Supervising projects of all sizes, from conception to completion – and typically simultaneously – will require a high degree of managerial ability

Process modelling

Process modelling involves graphically representing the functions, or processes, that capture, manipulate, store, and distribute data between a system and its environment and between components within a system. A common form of a process model is a data flow diagram (DFD).

The deliverables and outcomes of process modelling are:

1. Context DFD

2. DFDs of system (Adequately decomposed)

3. Thorough description of each DFD component

Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a graphical representation of the flow of information in a system. It shows the flow of data between external entities and the processes, data stores, and data flows within a system. DFDs are useful for modeling and analyzing the flow of information in a system, and they can be used at different levels of detail to represent different aspects of the system.

The various levels of DFD are:

1. Context Level DFD: This is the highest level DFD and shows the overall system boundaries and the relationship between the system and its external entities. It provides an overview of the entire system and is used to understand the overall flow of data.

There are several rules to follow when constructing a context diagram:

1. The context diagram should be simple and easy to understand

2. The context diagram should be a single diagram. It should not be broken down into multiple diagrams or levels.

3. Data stores should not appear in a context diagram.

4. It should show the main components of the system and the flow of data between them.

2. Level 1 DFD: This is the next level of DFD and shows the major processes within the system in more detail. It breaks down the system into smaller subsystems and shows how data flows between them.

3. Level 2 DFD: This level of DFD shows the processes from the level 1 DFD in even more detail. It breaks down the processes into smaller subprocesses and shows how data flows between them.

4. Level n DFD: This level of DFD shows the processes in the greatest level of detail. It breaks down the processes into the smallest possible subprocesses and shows how data flows between them.

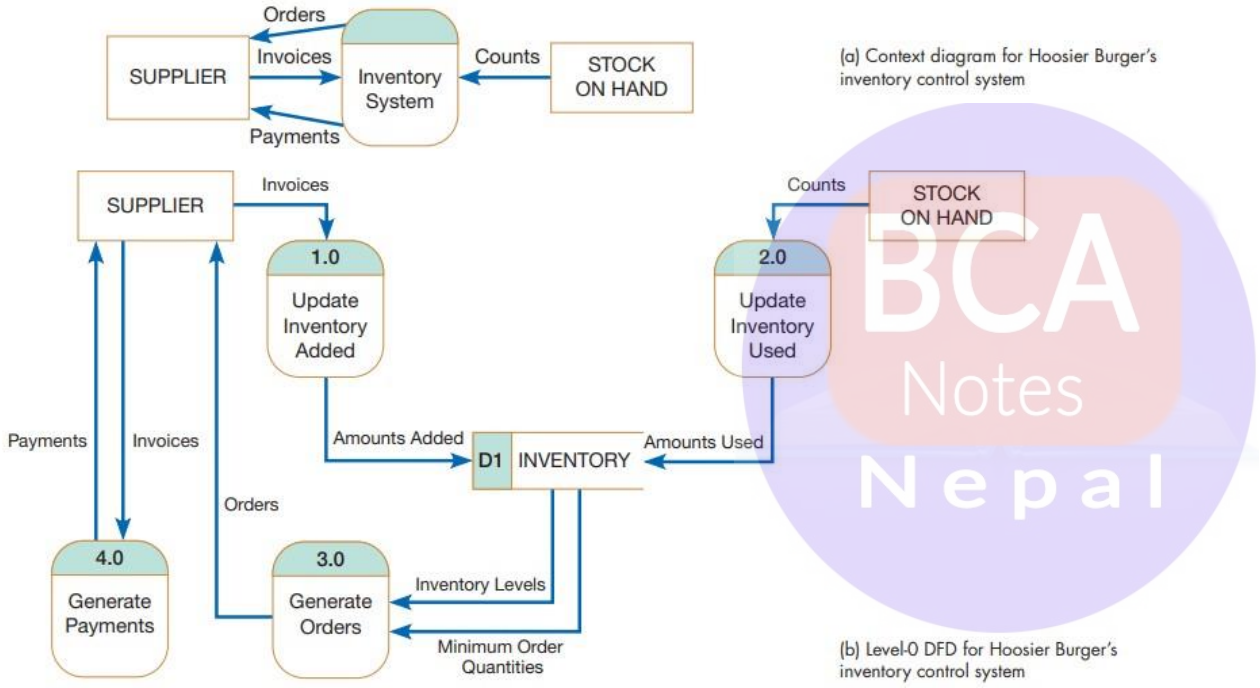

Fig: Example of a DFD

Logical modelling

Logical modeling is a method used to represent the relationships and constraints among variables in a problem or system using logical operators and logical equations. It is often used in computer science, artificial intelligence, and operations research as a way to formally describe and reason about problems, and to design and analyze algorithms for solving them. Logical models can be used to represent a wide range of problems and systems, including knowledge representation, planning, reasoning, and decision making. They can be implemented using various formal languages and tools, such as propositional logic, first-order logic, logic programming, and constraint programming.

Decision Tables

Decision tables are a graphical method of representing process logic. In decision tables, conditions are listed in the condition stubs, possible actions are listed in the action stubs, and rules link combinations of conditions to the actions that should result. Analysts reduce the complexity of decision tables by eliminating rules that do not make sense and by combining rules with different condition.

A decision table is called a process description tool because it provides a clear and structured way to describe the processes or rules that govern a system or problem. It describes how different inputs and conditions are related to different outputs and decisions, and how the system or problem should behave in different scenarios.

An example of decision table for payment system is given below. S stands for salary basis and H for hourly basis.

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce absence report		X				

Table: Decision Table

Conceptual data modeling

Conceptual data modeling is the process of creating a high-level representation of the data requirements of a system or organization. It involves identifying the main concepts, entities, and relationships that are relevant to the system or organization and representing them in a model. It is the first step in the data modeling process, and its main goal is to provide a clear and consistent understanding of the data requirements of the system or organization and ensure that the data model is aligned with the overall goals and objectives of the system or organization.

The primary deliverable from the conceptual data modeling step within the analysis phase is an E-R diagram. The other deliverable from conceptual data modeling is a full set of entries about data objects that will be stored in the project dictionary, repository, or data modeling software. The repository is the mechanism that links the data, processes, and logic models of an information system.

ER model and ER diagram

ER model (Entity-Relationship model) is a method used to represent the data requirements of a system or organization in a conceptual data model. It is used to capture the overall logical structure of the data and the relationships between the different entities and concepts, without specifying the physical structure or implementation details. The ER model is based on the idea that the data in a system can be represented as entities and the relationships between those entities.

An ER diagram (Entity-Relationship diagram) is a graphical representation of an ER model. It is used to visually represent the entities, attributes, and relationships in a system or organization. ER diagrams are composed of entities, which are represented as rectangles, and relationships, which are represented as diamonds. The entities and relationships are connected by lines, and attributes, which are characteristics of the entities, are represented as oval shapes connected to the entities.

ER diagrams are useful for designing and analyzing data structures and relationships in a system or organization. They can be used to identify the main entities, concepts, and relationships that are important to the system or organization, and to ensure that the data model is flexible and adaptable to changing requirements. They can also be used to communicate the data requirements with different stakeholders, such as developers, testers, and users, in a clear and concise way.

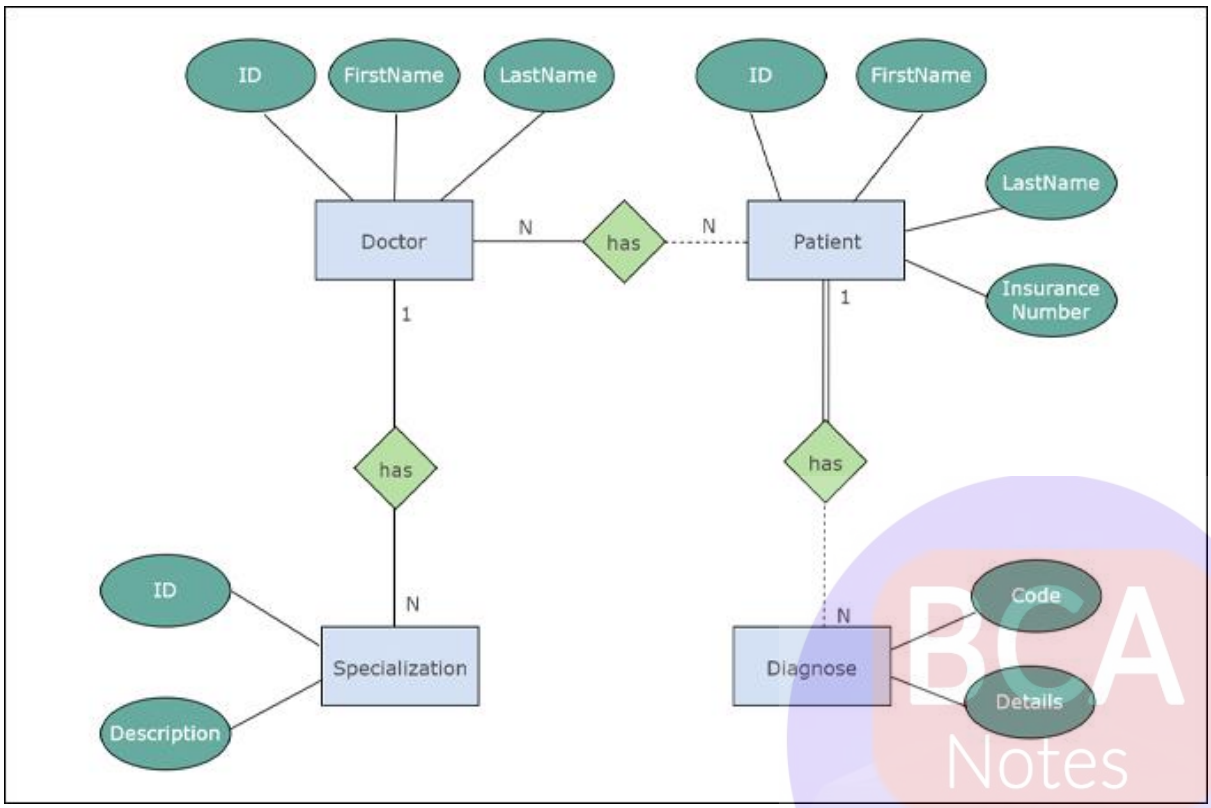

Fig: Example of ER Diagram

UNIT-4: Design

Database Design, Forms and Reports Design, Interface and Dialogue Design

Database Design

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. It is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. It helps to produce database system which have high performance and that can meet the requirements of the users. The main objective of database designing is to produce logical and physical designs models of the proposed database system.

Logical Database Design

A logical database is a conceptual representation of a database that describes the relationships between data entities and their attributes, without specifying how the data is physically stored. It is a way of organizing and structuring data in a way that makes it easy to understand, manage, and maintain. The logical database design is typically independent of a specific database management system (DBMS) and is used as a blueprint for physical database design.

The logical database is created through a process called logical database design, which involves several steps, including requirements gathering, conceptual data modeling, normalization, and refinement. The end result of the process is a well-organized and structured database that meets the needs of stakeholders, including business users, developers, and database administrators.

Physical Database Design

A physical database is the actual implementation of a database, where the logical design is translated into a specific database management system (DBMS) and optimized for performance. The physical database represents the storage of data on physical media, such as hard drives or solid-state drives.

Physical database design is the process of translating a logical database design into a specific database management system (DBMS) and optimizing it for performance. Physical database design involves several steps, including:

1. Mapping the logical design to the specific DBMS: This involves defining the data structures, such as tables, columns, and rows, as well as the relationships between them using keys, indexes, and constraints.
2. Storage allocation: Deciding on the physical storage of the data, taking into consideration factors such as storage capacity, data retrieval speed, and scalability.
3. Performance tuning: Optimizing the physical design to improve the speed and efficiency of data retrieval, insertion, and modification. This may involve using indexing, partitioning, and other techniques to improve query performance.
4. Security: Defining and implementing security measures to protect the data and control access to it, such as authentication, authorization, and encryption.
5. Data backup and recovery: Establishing procedures for backing up and recovering data in case of failures or other disruptions.

Relational Database Model

A relational database model is a way of organizing data in a structured format using tables, rows, and columns. The relational model is based on the mathematical concept of relations, which are a set of ordered pairs. In a relational database, data is organized into a set of tables, also known as relations, where each table represents a specific entity or concept.

Each table is made up of rows and columns, where a row represents a single record or data entry and a column represents a field or attribute of that record. The columns in a table are also known as attributes and have a specific data type, such as text, number, or date.

The relational model also uses a system of keys to link data between tables. A primary key is a unique identifier for each row in a table, and a foreign key is used to link data between two tables. For example, a foreign key in one table might reference the primary key of another table.

This allows the data in the tables to be related to one another, and to be queried and manipulated in a consistent and efficient way using SQL (Structured Query Language). The relational database model is widely used in many applications and it is the most common type of database in use today.

Normalization

Normalization is a process used in the design of relational databases to organize data into separate, related tables to minimize data redundancy and improve data integrity. It is a way of organizing data into multiple tables based on the relationships between the data. There are several normal forms (1NF, 2NF, 3NF, etc.) that provide a set of guidelines for organizing data in a relational database. The most commonly used normal forms are first normal form (1NF) and third normal form (3NF).

1NF: In 1NF, each table has a primary key and all data is atomic, which means that each field contains only one piece of data.

2NF: In 2NF, all non-key attributes are fully dependent on the primary key, which means that each field depends on the primary key to identify it.

3NF: In 3NF, there are no transitive dependencies. That means that no non-key fields are dependent on other non-key fields.

The importance of normalization in database design is that it helps to minimize data redundancy and improve data integrity. By organizing data into separate, related tables, normalization makes it easier to update, modify, and query the data without affecting other parts of the database. Additionally, it helps to avoid data inconsistencies and anomalies, such as update, insertion, and deletion anomalies.

Normalization also helps to improve performance and scalability, as it minimizes the amount of data that needs to be read and written to the database. Additionally, it makes it easier to maintain the database over time.

To normalize an unnormalized table, it is typically broken down into smaller, related tables. The process of normalizing a table typically involves moving data from a single, unnormalized table into multiple, related tables. The most common normal forms are 1NF, 2NF and 3NF.

1NF: To convert an unnormalized table to 1NF, each table must have a primary key, and all data must be atomic, meaning that each field contains only one piece of data. All repeating groups of data are removed and placed in separate tables. Each table must have a unique identifier (Primary Key) which is used to link the data back to the original table.

2NF: To convert a table to 2NF, all non-key attributes must be fully dependent on the primary key. This means that each field depends on the primary key to identify it. Any data that is not dependent on the primary key is removed and placed in separate tables.

3NF: To convert a table to 3NF, all data should not depend on any non-key fields. This means that there are no transitive dependencies. All data that is dependent on other non-key fields is removed and placed in separate tables.

File Organization

File organization refers to the way in which data is stored and retrieved in a file system. It is the process of arranging data in a specific way to optimize the efficiency and performance of file access operations. The main objectives of file organization are to:

1. Optimize file access time: By organizing files in an efficient way, it is possible to reduce the time required to access a file, which can improve the overall performance of the system.

2. Minimize disk space usage: By organizing files in an efficient way, it is possible to reduce the amount of disk space required to store the files, which can save resources and lower costs.

3. Facilitate data management: By organizing files in a logical and consistent way, it is possible to make it easier to manage and maintain the data, including operations such as backup, recovery, and archiving.

4. Improve data security: By organizing files in a specific way, it is possible to improve the security of the data by controlling access to the files and protecting them from unauthorized access.

5. Facilitate data sharing: By organizing files in a specific way, it is possible to make it easier to share data between different users and systems.

6. Improve data integrity: By organizing files in a specific way, it is possible to improve the integrity of the data by implementing error-checking and recovery mechanisms.

	File Organization		
Factor	**Sequential**	**Indexed**	**Hashed**
Storage space	No wasted space	No wasted space for data, but extra space for index	Extra space may be needed to allow for addition and deletion of records
Sequential retrieval on primary key	Very fast	Moderately fast	Impractical
Random retrieval on primary key	Impractical	Moderately fast	Very fast
Multiple key retrieval	Possible, but requires scanning whole file	Very fast with multiple indexes	Not possible
Deleting rows	Can create wasted space or require reorganizing	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy
Adding rows	Requires rewriting file	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy, except multiple keys with same address require extra work
Updating rows	Usually requires rewriting file	Easy, but requires maintenance of indexes	Very easy

Inverted List File Organization

Inverted list file organization, also known as inverted index, is a technique used in databases and information retrieval systems to efficiently store and retrieve data based on keywords or terms. It is a data structure that is used to map a set of keywords to the documents or records that contain them.

Forms and Reports Design

Designing forms involves the layout, design, and functionality of the forms, including the placement of fields, labels, and buttons, as well as validation rules and error handling. It's important to ensure that the forms are easy to use, visually appealing, and provide clear instructions to the users. The forms should also be designed to be responsive, so that they can be used on different devices and screen sizes.

Designing reports involves the layout, design, and functionality of the reports, including the selection of data fields, sorting and grouping of data, and the use of charts and graphics to display the data in a clear and meaningful way. The reports should be designed in a way that makes it easy to understand and interpret the data, with clear labels, headings and layout.

The main deliverables and outcomes of designing forms and reports include:

1. Form design: The layout, design, and functionality of the forms, including the placement of fields, labels, and buttons, as well as validation rules and error handling.

2. Report design: The layout, design, and functionality of the reports, including the selection of data fields, sorting and grouping of data, and the use of charts and graphics to display the data in a clear and meaningful way.

Interface and Dialogue Design

The process of designing interfaces and dialogues is a user-focused activity. This means that you follow a prototyping methodology of iteratively collecting information, constructing a prototype, assessing usability, and making refinements. Designing interfaces and dialogues requires understanding the user's needs and goals, as well as the capabilities of the system. The interface should be intuitive and easy to use, and should minimize the number of steps needed to accomplish a task. This can be achieved through the use of clear and consistent labelling, appropriate use of color and layout, and providing helpful feedback and error messages. The dialogue should be natural and easy to understand, and should provide clear and concise information. It should be designed to elicit the necessary information to complete the task and provide feedback to the user in a way that is easy to understand. To design usable interfaces and dialogues, you must answer the same who, what, when, where, and how questions used to guide the design of forms and reports. Thus, this process parallels that of designing forms and reports.

The deliverable and outcome from system interface and dialogue design is the creation of a design specification. It includes interaction methods and devices. Methods of interaction include command line interaction, menu interaction, form interaction, object-based interaction and natural language interaction. Designing layouts, Structuring data entry, controlling data input, etc. fall under interface design.

Methods of Interaction

1. Command Language Interaction In command language interaction, the user enters explicit statements to invoke operations within a system. This type of interaction requires users to remember command syntax and semantics. Command language interaction places a substantial burden on the user to remember names, syntax, and operations.

2. Menu Interaction A significant amount of interface design research has stressed the importance of a system's ease of use and understandability. Menu interaction is a means by which many designers have accomplished this goal. A menu is simply a list of options; when an option is selected by the user, a specific command is invoked or another menu is activated.

3. Form Interaction The premise of form interaction is to allow users to fill in the blanks when working with a system. Form interaction is effective for both the input and presentation of information. An effectively designed form includes a self-explanatory title and field headings, has fields organized into logical groupings with distinctive boundaries, provides default values when practical, displays data in appropriate field lengths, and minimizes the need to scroll windows.

4. Object-Based Interaction The most common method for implementing object-based interaction is through the use of icons. Icons are graphic symbols that look like the processing option they are meant to represent. Users select operations by pointing to the appropriate icon with some type of pointing device. The primary advantages to icons are that they take up little screen space and can be quickly understood by most users.

5. Natural Language Interaction One branch of artificial intelligence research studies techniques for allowing systems to accept inputs and produce outputs in a conventional language such as English. This method of interaction is referred to as natural language interaction. Presently, natural language interaction is not as viable an interaction style as the other methods presented.

Dialogue Sequence Design

First step in dialogue design is to define the sequence. In other words, you must first gain an understanding of how users might interact with the system. This means that you must have a clear understanding of user, task, technological, and environmental characteristics when designing dialogues. As a designer, once you understand how a user wishes to use a system, you can then transform these activities into a formal dialogue specification. A formal method for designing and representing dialogues is dialogue diagramming. Dialogue diagramming is a technique used in the design and development of dialogue systems, it is a visual representation of the flow of a conversation, it helps to understand and plan the system's functionality and to identify potential issues and challenges.

UNIT-5: Implementation and Maintenance

Implementation

Coding, Testing, Installation, Documentation, Training, Support

Software Application Testing

Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases. Software testing is a method of assessing the functionality of a software program. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability. It mainly aims at measuring specification, functionality and performance of a software program or application.

Advantages:

1. Cost Effective Development - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.

2. Product Improvement - During the SDLC phases, testing is never a time-consuming process. However, diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.

3. Test Automation - Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automaton should be started when the software has been manually tested and is stable to some extent. Moreover, test automation can never be used if requirements keep changing.

4. Quality Check - Software testing helps in determining following set of properties of any software such as Functionality, Reliability, Usability, Efficiency, Maintainability and Portability.

Types:

1. Inspections: A testing technique in which participants examine program code for predictable language specific errors. Syntax, grammar, and some other routine errors can be checked by automated inspection software. It has been estimated that code inspections detect 60 to 90 percent of all software defects as well as provide programmers with feedback that enables them to avoid making the same types of errors in future work.

2. Desk checking: Desk checking is an informal manual test that programmers can use to verify coding and algorithm logic before a program launch. This enables them to spot error that might prevent a program from working as it should. Modern debugging tools make desk checking less essential than it was in the past, but it can still be a useful way of spotting logic errors. It refers to the manual approach of reviewing source code (sitting a desk), rather than running it through a debugger or another automated process. In some cases, a programmer may even use a pencil and paper to record the process and output of functions within a program. For example, the developer may track the value of one or more variables in a function from beginning to end. Manually going through the code line-by-line may help a programmer catch improper logic or inefficiencies that a software debugger would not.

3. Unit testing: It focuses on the smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs. Automated technique whereby each module is tested alone in an attempt to discover any errors that may exist in the module's code.

4. Integration testing: The process of bringing together more than one module that a program comprises for testing purposes. Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

5. System testing: The process of bringing together of all of the programs that a system comprises for testing purposes. System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is alone with full system implementation and environment. Programs are typically integrated in a top-down incremental fashion.

The system can be tested in two ways:

I. Black Box Testing

i. Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program.

ii. This type of testing is carried out by testers.

iii. Implementation Knowledge is not required to carry out Black Box Testing

iv. Programming Knowledge is not required to carry out Black Box Testing.

v. Testing is applicable on higher levels of testing like System Testing, Acceptance testing.

vi. Black box testing means functional test or external testing.

vii. Black Box testing is primarily concentrated on the functionality of the system under test.

II. White Box Testing

i. White box testing is the software testing method in which internal structure is being known to the tester who is going to test the software.

ii. Generally, this type of testing is carried out by software developers.

iii. Implementation Knowledge is required to carry out White Box Testing.

iv. Programming Knowledge is required to carry out White Box Testing.

v. Testing is applicable on lower level of testing like Unit Testing, Integration testing.

vi. White box testing also means structural test or interior testing

vii. White Box testing is primarily concentrated on the testing of program code of the system under test like code structure, branches, conditions, loops etc.

6. Stub testing: A technique used in testing modules, especially where modules are written and tested in a top down fashion, where a few lines of codes are used to substitute for subordinate modules. Top-level modules contain many calls to subordinate modules, we may wonder how they can be tested if the lower level modules haven't been written yet. This is called stub testing.

7. User acceptance testing: Acceptance testing is testing the system in the environment where it will eventually be used. Acceptance refers to the fact that users typically sign off on the system and "accept" it once they are satisfied with it. The purpose of acceptance testing is for users to determine whether the system meets their requirements. The extent of acceptance testing will vary with the organization and with the system in question. Acceptance testing is used to evaluate the overall functionality, usability, and performance of the software, and to ensure that it meets the needs and requirements of the users. It typically includes testing the software's main features and functionality, as well as testing for usability, performance, and security.

It is of two types:

1. Alpha Testing
   Alpha testing refers to user testing of a completed information system using simulated data. During alpha testing, the entire system is implemented in a test environment to discover whether the system is overtly destructive to itself or to the rest of the environment. The types of tests performed during alpha testing include the following:
   i. Recovery testing—forces the software (or environment) to fail in order to verify that recovery is properly performed.
   ii. Security testing—verifies that protection mechanisms built into the system will protect it from improper penetration.
   iii. Stress testing—tries to break the system (e.g., what happens when a record is written to the database with incomplete information or what happens under extreme online transaction loads or with a large number of concurrent users).
   iv. Performance testing—determines how the system performs in the range of possible environments in which it may be used (e.g., different hardware configurations, networks, operating systems, and so on); often the goal is to have the system perform with similar response time and other performance measures in each environment.
2. Beta Testing
   Beta testing refers to user testing of a completed information system using real data in the real user environment. In beta testing, a subset of the intended users runs the system in the users' own environments using their own data. The intent of the beta test is to determine whether the software, documentation, technical support, and training activities work as intended. In essence, beta testing can be viewed as a rehearsal of the installation phase.

Software Installation

The process of moving from the current information system to the new one is called installation. In this activity, all the users must give up their reliance on the current system and begin to rely on the new system. There are four different ways of installation. The approach an organization decides to use will depend on the scope and complexity of the change associated with the new system and the organization's risk factor.

i. Direct installation: Changing over from the old information system to a new one by turning off the old system when the new one is turned on is called direct installation. Any errors resulting from the new system will have a direct impact on the users. If the new system fails, considerable delay may occur until the old system can again be made operational and business transactions are re-entered to make the database up to date. Direct installation can be very risky. Direct installation requires a complete installation of the whole system.

ii. Parallel installation: Running the old information system and the new one at the same time until management decides the old system can be turned off. All of the work done by the old system is concurrently performed by the new system. Outputs are compared to help determine whether the new system is performing as well as the old. Errors discovered in the new system do not cost the organization much, if anything, because errors can be isolated and the business can be supported with the old system.

iii. Pilot installation: It is also known as single-location installation. Rather than converting all of the organization at once, single location installation involves changing form the current to the new system in only one place or in a series of separate sites over time. The single location may be a branch office, a single factory, or one department, and the actual approach used for installation in that location may be any of the other approaches.

iv. Phased installation: It is also called staged installation, and is an incremental approach. With phased installation, the new system is brought online in functional components; different parts of the old and new systems are used in cooperation until the whole new system is installed. Phased installation, like single-location installation, is an attempt to limit the organization’s exposure to risk, whether in terms of cost or disruption of the business. By converting gradually, the organization’s risk is spread out over time and place. Also, a phased installation allows for some benefits from the new system before the whole system is ready.

Documentation

Documentation is the process of collecting, organizing, storing and maintaining a complete record of system and other documents used or prepared during the different phases of the life cycle of the system. System cannot be considered to be complete, until it is properly documented. Documentation is important in SDLC for the following reasons:

1. It solves the problem of indispensability of an individual for art organization as it provides a record of the software development process, including the requirements, design decisions, and testing results. Even if the person who has designed or developed the system, leaves the organization, the documented knowledge remains with the organization, which can be used for the continuity of that software
2. It makes the system easier to modify and maintain in the future. The key to maintenance is proper and dynamic documentation. It is easier to understand the concept of a system from the documented records.

Types:

1. System documentation: System documentation records detailed information about a system's design specification, its internal workings and its functionality. System documentation is intended primarily for maintenance programmers. It contains the following information:
   • A description of the system specifying the scope of the problem, the environment in which it functions, its limitation, its input requirement, and the form and type of output required.
   • Detailed diagram of system flowchart and program flowchart.
   • A source listing of all the full details of any modifications made since its development.
   • Specification of all input and output media required for the operation of the system.
   • Problem definition and the objective of developing the program.
   • Output and test report of the program.
   • Upgrade or maintenance history, if modification of the program is made.
   There are two types of system documentation. They are:
   i. Internal documentation: Internal documentation is part of the program source code or is generated at compile time.
   ii. External documentation: External documentation includes the outcome of structured diagramming techniques such as dataflow and entity-relationship diagrams.

2. User documentation:
   User documentation consists of written or other visual information about an application system, how it works and how to use it. User documentation is intended primarily for users. It contains the following information:
   • Set up and operational details of each system.
   • Loading and unloading procedures.
   • Problems which could arise, their meaning, reply and operation action.
   • Special checks and security measures.
   • Quick reference guides about operating a system in a short, concise format.

Maintenance

Obtaining maintenance requests, Transforming requests into changes, Designing Changes, Implementing changes

Once software system is put into use or installed, new requirements emerges and existing requirements change as the business running that system changes and the system is essentially in the maintenance phase of the systems development life cycle. When a system is in the maintenance phase, some person within the systems development group is responsible for collecting maintenance requests from system users and other interested parties, such as system auditors, data center and network management staff, and data analysts, etc.

There are four major activities occur within maintenance:

1. Obtaining Maintenance Requests: In this step a formal process is established whereby users can submit system change requests. When developing the procedures for obtaining maintenance requests, organizations must also specify an individual within the organization to collect these requests and manage their dispersal to maintenance personnel.

2. Transforming Request into Changes: Once a request is received, analysis must be performed to identify the scope of the request. It must be determined how the request will affect the current system and how long such a project will take. As with the initial development of a system, the size of a maintenance request can be analyzed for risk and feasibility.

3. Designing changes: Next, a change request can be transformed into a formal design change, which can then be fed into the maintenance implementation phase.

4. Implementing Changes: In this activity, once the change design is approved, proposed changes are implemented in respective components of the system.

Types:

1. Corrective maintenance: It refers to changes made to repair defects in the design, coding, or implementation of the system. This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumption Thus, in corrective maintenance, processing or performance failures are repaired.

2. Adaptive maintenance: In adaptive maintenance, program functions are changed to enable the information system to satisfy the information needs of the user. It involves making changes to an information system to evolve its functionality to changing business needs or to migrate it to a different operating environment. Within a home, adaptive maintenance might be adding storm windows to improve the cooling performance of an air conditioner.

This type of maintenance may become necessary because of organizational changes which may include:

a) Change in the organizational procedures

b) Change in organizational objectives, goals, policies, etc

c) Change in forms

d) Change in information needs of managers

e) Change in system controls and security needs, etc

3. Perfective maintenance: Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance undertaken to respond to riser's additional needs which may be due to the changes within or outside of the organization. Outside changes are primarily environmental changes, which may in the absence of system maintenance, render the information system ineffective and inefficient.

These environmental changes include:

a) Changes in governmental policies, laws, etc

b) Economic and competitive conditions, and

c) New technology

4. Preventive maintenance: Preventive changes refer to changes made to increase the understanding and maintainability of your software in the long run. Preventive changes are focused in decreasing the deterioration of your software in the long run. Restructuring, optimizing code and updating documentation are common preventive changes. Executing preventive changes reduces the amount of unpredictable effects software can have in the long term and helps it become scalable, stable, understandable and maintainable maintenance.